Lecture 23

Distribution of the quotient (q) values:

Code the non-negative integers n = mq + r, where m is a coding parameter (positive integer). Split the integer n into two parts:
(1) Code **q** with unary code. Here q is the quotient of (n/m).

Unary code: q 1's, followed by one '0'. Codeword length of this unary code: (q + 1) bits
Example, m = 16:

$m = 16$			
n	Codeword	n	Codeword
0	00000	24	101000
1	00001	25	101001
2	00010	26	101010
3	00011	27	101011
4	00100	28	101100
5	00101	29	101101
6	00110	30	101110
7	00111	31	101111
8	01000		
9	01001	32	1100000
10	01010	33	1100001
11	01011	34	1100010
12	01100	35	1100011
13	01101	36	1100100
14	01110	37	1100101
15	01111	38	1100110
		39	1100111
16	100000	40	1101000
17	100001	41	1101001
18	100010	42	1101010
19	100011	43	1101011
20	100100	44	1101100
21	100101	45	1101101
22	100110	46	1101110
23	100111	47	1101111

$n = 0, 1, 2, \cdots, 15$

$\frac{n}{m} = \frac{n}{16} \Rightarrow$ same q value = 0

$n = 16, 17, 18, \cdots, 31$

$\frac{n}{m} = \frac{n}{16} \Rightarrow$ same q value = 1

$\vdots$

In general:

For any give value of q, all these $n = mq + r$, where $r = 0, 1, \cdots, (m-1)$ generate the same q value.

Hence, $\text{Prob}[q] = \sum_{n=mq}^{mq+(m-1)} p^n (1-p)$

$$= (1-p) \sum_{n=mq}^{mq+(m-1)} p^n = (1-p)\left[\sum_{n=0}^{mq+(m-1)} p^n - \sum_{n=0}^{mq-1} p^n\right]$$

$$= (1-p)\,\frac{1 - p^{mq+m} - 1 + p^{mq}}{1-p}$$

$$= p^{mq}(1-p^m)$$

$$\sum_{n=0}^{K} p^n = \frac{1-p^{K+1}}{1-p}$$

In summary,

$\text{Prob}[q] = p^{mq}(1-p^m)$, where $q = 0, 1, 2, \cdots$ (non-negative integers)

Let $s = p^m$, then $\text{Prob}[q] = s^q(1-s)$,

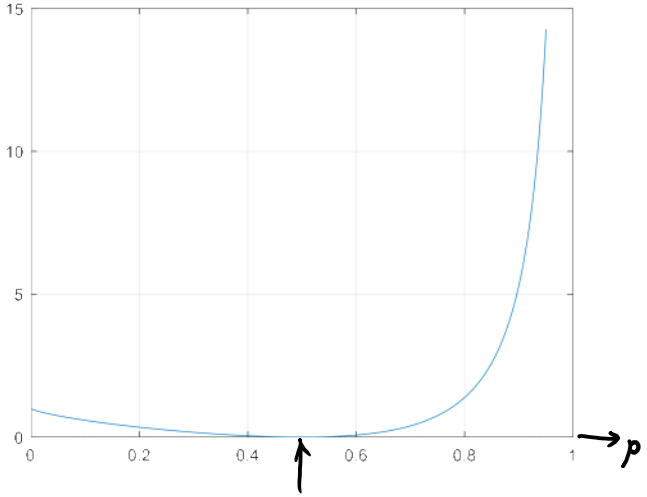
Geometric Distribution

Golomb code: $\boxed{p^m = \frac{1}{2}} \Rightarrow s = \frac{1}{2} \Rightarrow \text{Prob}[q] = \left(\frac{1}{2}\right)^q\left(\frac{1}{2}\right) = \left(\frac{1}{2}\right)^{q+1}$

(1) Code **q** with unary code => coding efficiency being optimal = entropy of q.

$G(n) = \left(\frac{1}{2}\right)^{n+1}$

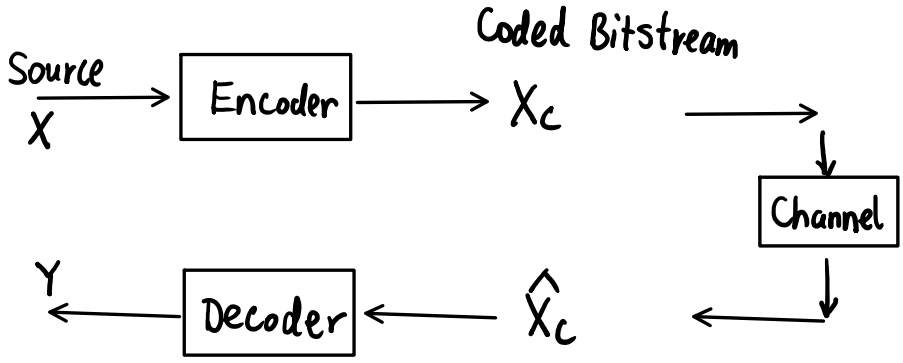
n	$G(n)$	$m = 1$ Codeword
0	1/2	0
1	1/4	10
2	1/8	110
3	1/16	1110
4	1/32	11110
5	1/64	111110
6	1/128	1111110
7	1/256	11111110
8	1/512	111111110
9	1/1024	1111111110
10	1/2048	11111111110



$p = \frac{1}{2}$, Redundancy = 0.

In this special case, the Golomb code becomes the unary code.

- Lossy Coding



Distortion:

$$D = E\left[(X-Y)^2\right]$$

where $X = (X_1, X_2, \cdots, X_N)$, $Y = (Y_1, Y_2, \cdots, Y_N)$

$$D = \frac{1}{N}\sum_{i=1}^{N}(X_i - Y_i)^2$$

JPEG Lossy Image **Compression** using DCT

- DCT (Discrete Cosine Transform)
  Good for concentrating input signal energy.

Discrete Cosine Transform
The discrete cosine transform (DCT) is closely related to the discrete Fourier transform. You can often reconstruct a sequence very accurately from only a few DCT coefficients. This property is useful for applications requiring data reduction.

```
>> X
X =
   1    2    3    4

>> doc dct
>> Y = dct(X)
Y =     DCT coefficients
  5.0000   -2.2304         0   -0.1585
```

```
>> sum(X.^2)
ans =
   30:   Input signal energy

>> sum(Y.^2)
ans =
  30.0000:   DCT output signal energy
```

Y = DCT coefficients

5.0000 -2.2304 0 -0.1585

$\frac{5^2}{30} = \frac{25}{30} = \frac{5}{6} \doteq 83.3\%$

Keeping the first two coefficients: 99.92%

30.0000: DCT output signal energy

```
>> (Y(1)^2 + Y(2)^2)/sum(Y.^2)
ans =
    0.9992
```

```
>> Yhat(3:4) = 0;
>> Yhat
Yhat =
   5.0000   -2.2304         0         0
```

```
>> Xhat = idct(Yhat)
ans =
   1.0429    1.8964    3.1036    3.9571
```

```
>> sum(Xhat.^2)
ans =
   29.9749
```

```
>> sum(Xhat.^2)/sum(X.^2)
ans =
   0.9992
```
⇒ 99.92%

Distortion between X and Xhat:

```
>> sum((X - Xhat).^2)
ans =
   0.0251
```

```
Y =    DCT coefficients
   5.0000   -2.2304         0   -0.1585
```

```
>> Y(3).^2 + Y(4).^2
ans =
   0.0251
```
dropped coefficients

Or use the norm ( ) function on Matlab:

```
>> (norm(X - Xhat))^2
ans =
   0.0251
```

$$D = \frac{1}{N} \sum_{i=1}^{N} (X_i - Y_i)^2$$

```
>> (norm(X - Xhat))^2/4
ans =
   0.0063
```